

CS 203: Assignment #2

Calculator

Write a program to simulate a simple Calculator. It accepts an arithmetic expression, evaluates it and then displays the result.

The expression will be entered in an **infix** form. For ease of computation, the mathematical expression should be converted into **postfix** form.

Steps to convert infix form into postfix form (using a stack – operator stack):

1. Read an expression (infix form).
2. Each time you find a left parentheses, push it into the stack.
3. Each time you find a right parentheses, pop all operators from stack and store them in the postfix until left parentheses is encountered.
4. Each time you find a number, put it in postfix.
5. Each time you find an operator :
if the new operator has less precedence or equal to the old operator (in stack top) then pop an operator and put it in postfix then push the new one. Otherwise, push the new operator into stack.
6. When the expression finished, pop all operators and put them in postfix.

Exp:

If you have this expression: $(6+2)*5-8/4=$

Expression in postfix: $62+5*84/-=$ How??

Steps to compute postfix expression (using stack – operand stack):

1. Each time you find a digit push it into the stack.
2. Each time you find an operator, pop two digits from stack top and compute the result.
3. Push this result into the stack.

Exp:

If this is the postfix $62+5*84/-=$, the result is 38

Notes:

1. The mathematical expression is a string.
2. Deal with the basic operators: $+ - * /$
3. The stack structure:

class Stack

{

int top;

char stackarr[20];

isEmpty //func. To check if stack is empty

isFull //func. To check if stack is full

public:

Stack //zero argument constructor

Push //func. To add an element to the stack, if possible

Pob //func. To delete an element from the stack, if possible

operator stack

operand stack